Discovery and Data Preparation

Andrew Nguyen

October 11, 2019

Introduction

This portfolio project is motivated mainly by self-driven curiousity as well as current interests. I have always been a gamer since childhood, but I am interested mainly in how video games are doing from a business standpoint. On average, I may spend over $100 on certain microtransactions in a couple of mobile games and may dedicate several hours each night on them. In particular, I am interested in the presence of microtransactions (in-app purchases) and how popular/successful games are with such.

I originally wanted to research the declining birth rates occuring globally over the years under the “Life and Death” category. However, I found that there are not enough data sets that are considerable for the purposes of this class, and my interest waned with the subject in addition.

Dataset

My initial dataset is a comma-separated value table consisting of a list of Apple App Store mobile games relevant data including Average User Rating, User Count, etc. It consists of 18 columns (10 of which I currently consider categorical, and 8 continuous) with 17,707 games documented.

Source

The data set came from a Kaggle posting. Its user, tristan581, collected the data on August 3, 2019 using iTunes API and the Appstore Sitemap. It can be found here. (https://www.kaggle.com/tristan581/17k-apple-app-store-strategy-games)

The data comes directly from the parent company themselves (Apple Inc) and thus the data here can be considered reliable and potentially pristine (meaning not touched by multiple other entities besides the company and dataset author). However, said data was not officially gathered by the company, and a potential threat would be the user directly having a hand in modifying some aspects of the data set.

Variables

1. **URL**: a URL in which anyone can access to view and download the game.
2. **ID**: unique ID assigned to each game for identification.
3. **Name**: the name of the game for easy human identification.
4. **Subtitle**: subtitle that usually accompanies a game title (if given).
5. **Icon URL**: 512px x 512px jpg link of the currently-used app icon.
6. **Average User Rating**: Rounded to nearest .5, requires at least 5 ratings
7. **User Rating Count**: Total number of ratings internationally, null means it is below 5.
8. **Price**: In USD.
9. **In-app Purchases**: Prices of microtransactions (if they exist)
10. **Description**: App description of what playing it usually entails.

11. **Developer**: The Company / Author / Developer of the game.
12. **Age Rating**: Categorized in 4+, 9+, 12+ or 17+.
13. **Languages**: ISO2A Language codes used in the game.
14. **Size**: Size of the app in bytes.
15. **Primary Genre**: Primary genre of the app
16. **Genres**: Specific genres of the game.
17. **Original Release Date**: When the app was released. DD/MM/YYYY.
18. **Current Version Release Date**: When the app was last updated. DD/MM/YYY.

Data Manipulation

First, we get our data and load it into R Studio.

```
# Here, I read the .csv file and store it as a dataset called InitialGameData. This will be the initial, uncleaned version of data I will clean later. The 'header' argument reads the first row as the dataset's variables, whereas the 'sep' argument indicates what character is separating each field; in our case, this is a comma-separated value file hence ",".
InitialGameData <- read.csv("appstore_games.csv", header=TRUE, sep=",")

# Use View to literally view the data set we just made.
View (InitialGameData)

# And run a few other functions for facts and kickers
ncol(InitialGameData) #should return 18 columns
```

```
## [1] 18
```

```
nrow(InitialGameData) #should return 17007 observations/rows
```

```
## [1] 17007
```

```
# Now that we have our initial but unclean data set, we need to "tidy" it now. In case we make mistakes and data goes bonkers or missing, however, we should make an identical version of the data set that we can freely operate on. InitialGameData is merely our "initial" data that will act as backup.
GameData <- InitialGameData
#View (InitialGameData)
```

Now, we work to tidy the data. We will need to load some libraries first, both for cleaning and for visualization purposes later.

```
#load libraries to properly plot data
library(ggplot2)
library(sjPlot)
```

```
## Learn more about sjPlot with 'browseVignettes("sjPlot")'.
```

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────────────────────────────────────── tidyverse 1.2.1 ──
```

```
## ✔ tibble  2.1.3     ✔ purrr   0.3.2
## ✔ tidyr   1.0.0     ✔ dplyr   0.8.3
## ✔ readr   1.3.1     ✔ stringr 1.4.0
## ✔ tibble  2.1.3     ✔ forcats 0.4.0
```

```
## ── Conflicts ────────────────────────────────────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(forcats)
library(plyr)
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following object is masked from 'package:purrr':
##
##     compact
```

We can see that the table is already pretty clean. However, we can make the column names a bit more palateable, shorter, or more descriptive. In particular, we replace periods with whitespace on column names for more organic human reading.

```
colnames(GameData)[colnames(GameData)=="Average.User.Rating"] <- "Average User Rating"  #colnames first takes in the data set name, then specify an existing column name and give it a new handle after "<-".
colnames(GameData)[colnames(GameData)=="User.Rating.Count"] <- "Total Ratings" #a more accurate name for this column in my opinion
colnames(GameData)[colnames(GameData)=="Icon.URL"] <- "App Icon URL"
colnames(GameData)[colnames(GameData)=="Age.Rating"] <- "Age Rating" #replace period
colnames(GameData)[colnames(GameData)=="In.app.Purchases"] <- "In-App Purchases" #syntax fix
colnames(GameData)[colnames(GameData)=="Primary.Genre"] <- "Primary Genre" #replace period
colnames(GameData)[colnames(GameData)=="Genres"] <- "Game Genres" #more descriptive
colnames(GameData)[colnames(GameData)=="Original.Release.Date"] <- "Original Release Date" #replace periods
colnames(GameData)[colnames(GameData)=="Current.Version.Release.Date"] <- "Current Version Release Date" #replace periods
```

Several variables in this data set should be set as factors (or organizable categorical data). In particular, Age Ratings are organized into 4+, 9+, 12+, and 17+. More on these can be read here (https://www.bewebsmart.com/ipod-ipad-iphone/how-to-restrict-apps-by-rating/). Other factor variables of note are Languages, In-App Purchases, Primary Genre, and Game Genres. We should check to see if they already are categorized as factors or not.

```
is.factor(GameData$`Age Rating`) # this should return true, indicating it's already implemented as a factor variable.
```

```
## [1] TRUE
```

```
is.factor(GameData$`Languages`) # should be true
```

```
## [1] TRUE
```

```
is.factor(GameData$`In-App Purchases`) # should be true
```

```
## [1] TRUE
```

```
is.factor(GameData$`Primary Genre`) # should be true
```

```
## [1] TRUE
```

```
is.factor(GameData$`Game Genres`) # should be true
```

```
## [1] TRUE
```

```
# Here, we run levels on each of the factor columns we checked to see what exactly the levels of each factor
 there are.

#levels(factor(GameData$`Age Rating`))  #4+, 9+, 12+, and 17+
#levels(factor(GameData$`In-App Purchases`)) #various prices
#levels(factor(GameData$`Languages`))  #various languages
#levels(factor(GameData$`Primary Genre`)) #Games and Entertainment primarily, can branch to other genre
s
#levels(factor(GameData$`Game Genre`))  #Same as above
```

The table is cleaner than we thought. At this point, we can simply organize the data into smaller tables.

```
#Here we create the Ratings table, which consists of the Game ID, Average User Rating, Total number of Ratin
gs, and Age Ratings. Note we have to use underscores rather than whitespace in column naming for successf
ul tibble compiling, but we're not gonna fuss over that.
Ratings <- tibble(ID = GameData$ID,
            Average_User_Rating = GameData$`Average User Rating`,
            Total_Ratings = GameData$`Total Ratings`,
            Age_Ratings = GameData$`Age Rating`)
#View(Ratings) #should return a successfully-made table

#Now, we want to create a table with only games with microtransactions. We will include variables from Ratin
gs and see if the value of the ratings corresponds with the presence of microtransactions.

nrow(GameData)  #should return 17007
```

```
## [1] 17007
```

```
gameswithpurchases <- GameData[-which(GameData$`In-App Purchases` == ""), ]  #collects data and exclu
de entries with no value in its In-App Purchases section.
nrow(gameswithpurchases)  #should return less than 17007; in our case, it will be 7683
```

```
## [1] 7683
```

```
GamePurchaseWithRatings <- tibble(ID = gameswithpurchases$ID,
                    Average_User_Rating = gameswithpurchases$`Average User Rating`,
                    Age_Ratings = gameswithpurchases$`Age Rating`,
                    InApp_Purchases = gameswithpurchases$`In-App Purchases`)

# For example plotting purposes, we can also make a table to see if the size of a game correlates to its rating. I
nstead of making a new table, we can add the column to the existing dataset using the cbind function (Take
 a dataset and specify the name of the new column as well as source data).

Ratings <- cbind(Ratings, Size = GameData$Size)
```

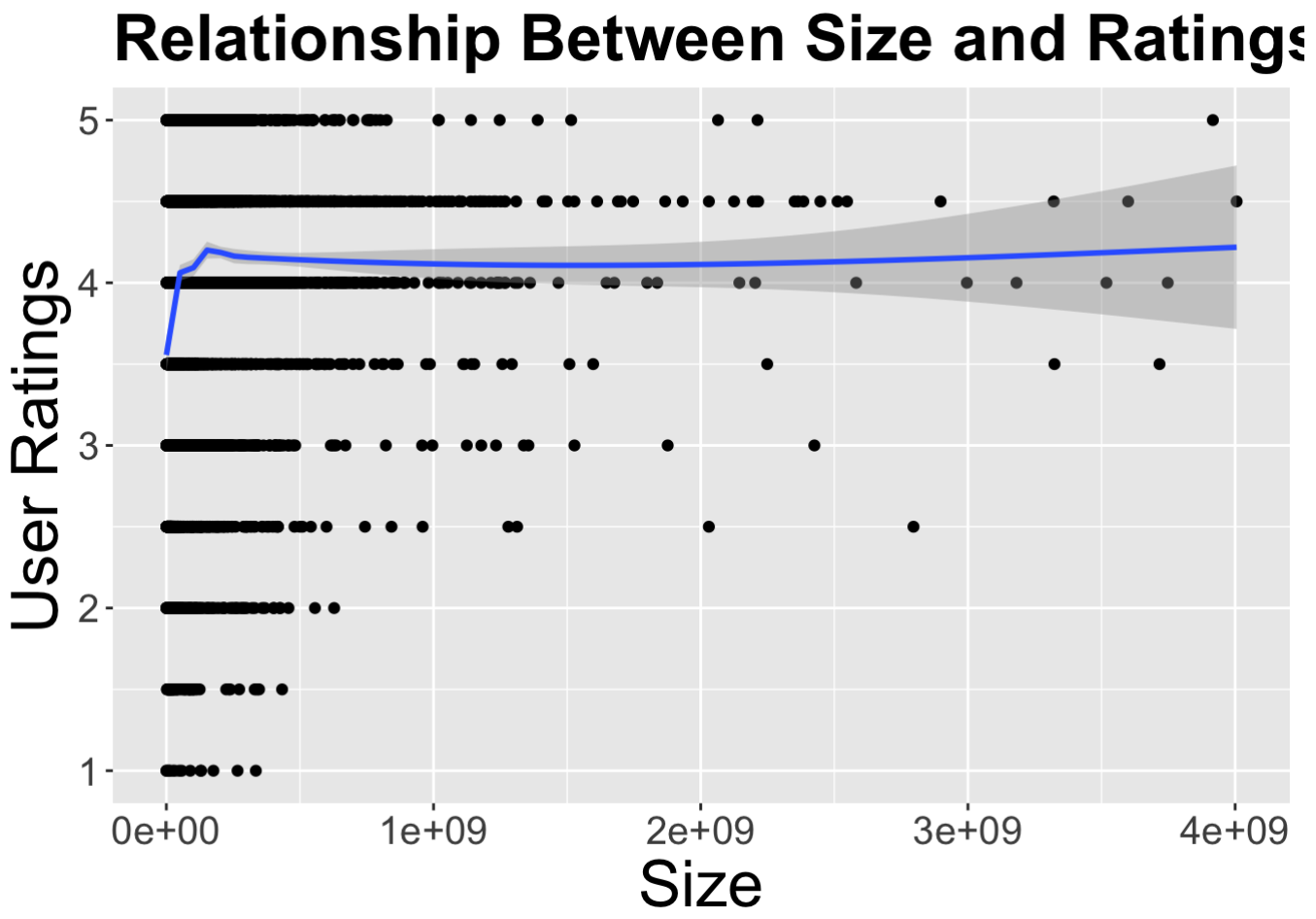
Visualizations

```
# For my first visualization, I take the example plotting we established last time and attempt to see the relations
hip between the size of a game and its ratings.
ggplot(Ratings, aes(x=Size, y=Average_User_Rating)) + geom_point() + geom_smooth() + labs(title="Relationshi
p Between Size and Ratings", x="Size", y="User Ratings") + theme(plot.title=element_text(size=25, face="bold"),
          axis.text.x=element_text(size=15),
          axis.text.y=element_text(size=15),
          axis.title.x=element_text(size=25),
          axis.title.y=element_text(size=25))
```

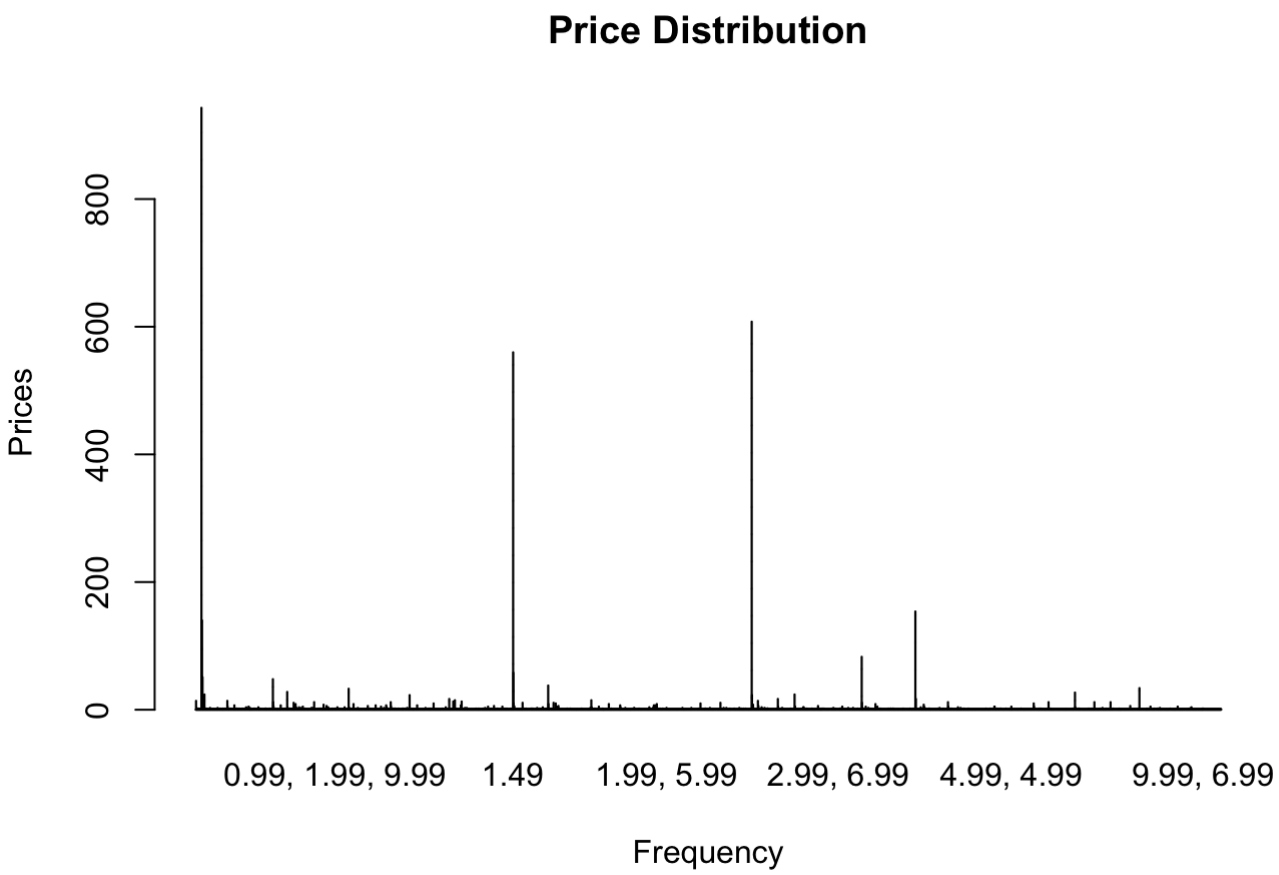
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

```
## Warning: Removed 9446 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 9446 rows containing missing values (geom_point).
```




```
# We can also view a glance of the frequencies of prices used as microtransactions:
counts <- table(GamePurchaseWithRatings$InApp_Purchases)
PriceDistribution <- barplot(counts, main="Price Distribution", width = 100, xlab = "Frequency", ylab = "Prices", ax
es = TRUE, density = 10)
```



Potential Research Questions

Some research questions unanswered here are as follows:

1. Do games with in-app purchases or a price correspond to an age group? (for example, are apps intended for 4+ usually free?)
2. Do games with in-app purchases or a price necessarily have a lower or higher rating than those without?
3. Does the rating of a game correspond with the age-group? (for example, are apps intended for 4+ usually ranked higher than those 17+ as users in the latter are more critical of what they play?)
4. Do games with more supported languages have a better rating turnout?
5. Are apps by certain developers more highly-rated than others?

Conclusion

This data set has neat potential for further analysis, especailly given the price, in-app transactions, ratings, and total ratings columns. Despite the threat of data tampering, I believe there is little reason to do exactly that on a mere datascrape. It would be interesting if the dataset also provided other areas of data, such as previous versions (which would raise questions such as: are games that are continuously updated usually rated highly?).